Analysis of User Comments: An Approach for Software Requirements Evolution

Laura V. Galvis Carreño and Kristina Winbladh
Department of Electrical and Computer Engineering
University of Delaware
Newark, DE USA
{lgalvis,winbladh}@udel.edu

***Abstract*—User feedback is imperative in improving software quality. In this paper, we explore the rich set of user feedback available for third party mobile applications as a way to extract new/changed requirements for next versions. A potential problem using this data is its volume and the time commitment involved in extracting new/changed requirements. Our goal is to alleviate part of the process through automatic topic extraction. We process user comments to extract the main topics mentioned as well as some sentences representative of those topics. This information can be useful for requirements engineers to revise the requirements for next releases. Our approach relies on adapting information retrieval techniques including topic modeling and evaluating them on different publicly available data sets. Results show that the automatically extracted topics match the manually extracted ones, while also significantly decreasing the manual effort.**

***Index Terms*—User feedback, Requirements, Topic Modeling, User Comments, Software Evolution, Information Retrieval.**

I. Introduction

User feedback is imperative in improving software quality. Many software companies collect data on user satisfaction through various means including focus groups, surveys, and error reports. Software users can also express their opinions on software applications in comments published on sites where an application is available for download. This type of software distribution is particularly common on platforms that offer a wide range of third party developed applications such as the Android Marketplace. The user feedback available on distribution sites thereby supply software companies with a rich source of information that can be used to improve future releases. In fact, user feedback can be and has been the driving force in software evolution of these kinds of applications. It is not uncommon to see software releases on platforms such as Apple's App Store state that the newest release addresses many of the issues raised by users. Common challenges in user-driven software evolution include a short time-to-market and obtaining the set of requirements that drive the changes. In this paper we address these two challenges by offering an automated analysis technique of user feedback. The technique adapts information retrieval (IR) techniques to extract common topics and present users' opinions about those topics. The goal of the work is to focus software developers on the most important requirements changes per user feedback at a lower effort level compared to manual processing of the feedback.

Fig. 1 provides a high-level overview of our approach to automatically extract topics for changing and creating new requirements that better represent user needs. The technique extracts topics prevalent in user comments and outputs results to be analyzed by the software team.

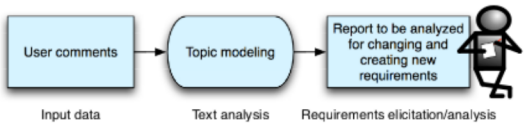

Fig. 1. High-level overview of approach.

We applied this approach on a set of user comments for three publicly available applications on the Android Marketplace. We evaluate the results by: (1) comparing the effort of manual and automated extraction of requirements from the user comments, (2) computing the accuracy of the results obtained with the automated technique compared to the manual results (which are used as a true set), and (3) reasoning about the possible requirements changes. As we are not the developers of the applications we use as input, we cannot establish a true set that is undoubtedly true. To avoid bias, we establish the true set manually prior to running our approach.

The results show that the approach is successful in classifying different topics present in the data sets. When compared to the manually generated data sets, we have levels of precision around 90% and recall values ranging from 50-80% depending on the number of topics we specify in the algorithm. With regard to requirements changes, we see that many of the topics extracted lead to both new and changed requirements.

The remainder of the paper is organized as follows: in Section II, we introduce related work; Section III describes our approach in detail; Section IV describes the experiments, and Section V summarizes the results of our evaluation; and finally, we conclude and outline future research in Section VI.

II. Related Work

Current studies in requirements evolution mainly focus on two aspects: requirements evolution during the life cycle of one version of a software and requirements evolution among different versions of a software product. To support the second aspect, researchers have analyzed the relationship between requirements, requirements evolution, and defects [1]. The

978-1-4673-3076-3/13/$31.00 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

outcome shows that there is no strong correlation between the number of requirements and the number of evolved requirements, and more interestingly that the number of defects is more strongly correlated with the number of evolved requirements than the total number of requirements. Identifying and documenting requirements changes is thus very important with regard to defect estimation as it can be more accurately determined if based on the evolved, and in particular added requirements, rather than on the total number of requirements.

The analysis of user satisfaction is helpful in improving products in general and is already an integral part of requirements evolution. However, using user satisfaction models based on time-consuming surveys becomes inadequate for software products that need to evolve rapidly not lose market share and user interest. Recent work in RE addresses this shortcoming by suggesting to use a mobile feedback system that allows users to provide guided feedback spontaneously as they use a software application [2]. The technique suggests a way to focus and filter the feedback to make it usable by software developers while maintaining relevance and ease of submission for users. Key to this approach is a classification of topics and sentiments. The obvious drawback of pre-defining entities is mitigated by allowing users to specify new entities. The results show that users came up with a significant number of entities that the software stakeholders had not considered.

Other work includes more automated approaches, particularly in data mining, to quickly extract user feedback and avoid the shortcomings of surveys. Topic modeling, including unsupervised analysis and automatic inference of semantic topics, has previously been used to extract opinions from text corpora such as Twitter, YouTube, and blogs and the results have been used to suggest other relevant items (movies/videos, people, events, and groups) to the user [3], [4].

In software engineering, researchers have applied topic modeling to various aspects of software projects, including traceability to follow the life of requirements [5], source code to help support understanding and find reusable components [6], and documentation to explore the evolving stream of topics over time [7]. IR techniques have also been used to locate concepts in source code. The underlying idea of this technique is to treat source code as a text corpus, use IR methods to index the corpus, and then build a search engine. The search engine allows developers to write textual queries based on their change request and the engine then retrieves relevant code documents [8] and clusters of code documents [9]. Another software engineering application focuses on comments posted in bug reports to locate and fix bugs, here, the goal is to measure the textual coherence and relevance of user comments in bug reports, which should contain good textual descriptions of the problem and properly selected attributes [10].

An approach closely related to ours is the work by Li et al. [11] that compares user satisfaction before and after software evolution with the goal of providing instructive information for the next step. The work suggests that developers should pay attention to three kinds of indicators. First, *why does the satisfaction in any indicator, e.g., operability, significantly change between versions, especially negatively?*. Here, developers find the reasons for the changes to develop the next version. Second, *which indicators get high degrees of user concern?*, such indicators are more important because they are what users are concerned about. Third, *which indicators that developers put much effort into present only small changes in user satisfaction?*, this provides information about the effectiveness of the developers' effort. Our work can complement these questions, but focuses more on deriving topics of interest and correlating those with requirements changes.

Liu et al. presents an approach that helps developers judge the quality of a software based on searching and analyzing comments on the Internet [12]. Here, the sentiment polarity, positive or negative, is identified and the quality aspects, which are described in the comments are extracted. This work has great applicability in the process of software resource selection and reuse. The main difference with our work is the interest in a particular feature, instead of all the possible features or topics. An example of a feature of interest of their work is the quality of the resources, and to study it, they evaluate, for example, the organization, flexibility, and stability.

Another interesting body of work includes automated sentiment analysis, which has been applied in different domains mainly outside software engineering. Brody used sentiment analysis to determine the general attitude of restaurant patrons posting reviews online [13] and Pal [14] to perceive customer sentiments toward products. However, using only sentiment analysis or topic modeling alone does not provide the association between topics and attitudes, which is crucial for informing software developer about requirements changes.

To address the joint problem of discovering the topics in reviews and the sentiments toward those topics, recent work has applied a combination of topic modeling and sentiment analysis to reviews of electronic devices and restaurants [15] as well as movies [16]. These efforts extend Latent Dirichlet Allocation (LDA), one of the most popular topic models, based on the assumption that documents are a mix of topics, where a topic is a probability distribution of words.

Our work adapts the model proposed by [15], Aspect and Sentiment Unification Model (ASUM), which incorporates both topic modeling and sentiment analysis and we use the model on user reviews of software applications. We chose this technique based on the possibility of associating topics with sentiments. Although, in this work, we describe just the first step towards this goal, i.e., topic extraction, as an initial assessment of the validity of using topic modeling to get information relevant to requirements evolution. We adapt the ASUM approach, as described in the subsequent section, to better fit in the domain of user feedback on software applications and to extract topics that are relevant to requirements engineers.

III. Approach

We use Information Extraction (IE) techniques [17] including topic modeling to automatically get constructive feedback from user comments. The goal is to find information that

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

can be used to change and/or create new requirements for a future release of a software. We take user comments under the premise that users would assess a software application through their opinions in user comments. An underlying assumption of our approach is that the importance of a topic is proportional to the number of comments it receives. The set of comments is processed to extract different topics that users have identified as worth commenting on. These topics will be different depending on the software application. Finally, a set of topics with a set of representative comments is produced. This set should be used by software developers to identify new and changing requirements in a similar way that they would use survey data and other user feedback data. Fig. 2 presents an overview of this approach as three main steps.

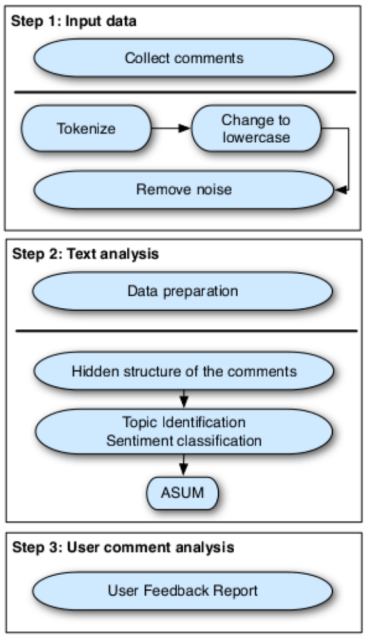

Fig. 2. User comments analysis approach

A. Step 1: Input Data

The first step includes the acquisition and preparation of data. In our experiments, (see Section IV), we use data from several mobile applications from the Android Marketplace. The general approach is the same if data was obtained elsewhere. The data then undergoes several pre-processing steps.

1) Tokenize: Each comment consists of a number of sentences or phrases. Tokenization converts a stream of characters into a sequence of tokens. A token is generally a word, but could also be a paragraph, a sentence, a syllable, or a phoneme. We separate the words by removing white spaces, punctuations, brackets, and other common sentence delimiters.

2) Change to Lowercase: We change each word into lowercase, in order to compare them more easily. Our approach does not treat nouns and proper nouns differently and therefore the change does not affect the data.

3) Remove Noise: The textual input contains noise, e.g., there are many words in the comments that are not helpful with regard to topic and sentiment extraction. We first apply a filter to remove non-words and non-numerical characters. We then remove words that match very common English words, so called *stop words*. The stop words typically include numbers, individual characters, articles, and conjunctions. We use the list provided and used in [18], because it includes individual characters and standard words with the same root. Prior to removing the stop words, we analyze sentences for negation. For example to classify "not good" as a negative sentiment, it is necessary to consider the two words collectively. We therefore put the "not" prefix to a word when the word is located closely (one or two words) behind a "not" as in [19]. Finally, we also remove comments that have become empty as a result of the noise reduction. The original textual data is thereby converted into data with no punctuation and stop words. In addition, words and sentences with less than three characters or words are removed. These filters reduce the input data significantly to only contain the most relevant words to be classified.

B. Step 2: Text Analysis

The pre-processing steps conducted on the input data result in a vector $P = \{s_1, s_2, ..., s_M\}$ where M is the number of sentences after noise reduction. We then analyze the sentences to find topics associated with each sentence. Here, we create two special data structures commonly used in IR. We can then apply the Aspect and Sentiment Unification Model (ASUM) approach on these structures to receive the classification.

1) Data Preparation: From P we create two data structures: (1) a word list W containing all the unique words of the collection, and (2) a bag of sentences B where each sentence is composed of indices referencing the words in the word list.

2) Hidden Structure of the Comments: We use topic modeling with the hypothesis that the user comments in the collection share the same set of topics and each comment exhibits those topics with different proportion. The original ASUM approach uses sentiment analysis to obtain knowledge about user opinions [4]. The goal is to use the implicit knowledge to extract an approximation of the hidden structure of the comments and sentiments [20]. The ASUM approach is applied to the entire collection of pre-processed data. ASUM [15] is an extension of Latent Dirichelt Allocation LDA [21] and constrains the words in a sentence to come from the same language model, and includes sentiment classification.

The graphical representation, or plate notation, of ASUM is shown in Fig. 3. The nodes are random variables, the directed edges between variables are dependencies, and plates are replications. Here only shaded nodes are observable, the other variables are latent. The variables ϕ, θ, and π are the words, topics, and sentiment distributions. The variables α, β, and γ are the parameters of the Dirichlet prior for θ, ϕ, and π. The sentiment, topic, and word are represented by s, z, and w. The representation of the number of sentiments, sentences, topics, and words are S, M, T and W. The words are the only observable variables, and the Dirichlet prior parameters are assigned at the start of the process. The latent variables are inferred by Gibbs sampling. The process is as follows:

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

1) For every pair of sentiment s and aspect z, draw a word distribution $\phi_{sz} \sim Dirichlet(\beta_s)$
2) For each comment d,
   a) Draw the comment's sentiment distribution $\pi_d \sim Dirichlet(\gamma)$
   b) For each sentiment s, draw an aspect distribution $\theta_{ds} \sim Dirichlet(\alpha)$
   c) For each sentence,
      i) Choose a sentiment $j \sim Multinomial(\pi_d)$
      ii) Given a sentiment j, choose an aspect $k \sim Multinomial(\theta_{dj})$
      iii) Generate words $w \sim Multinomial(\phi_{jk})$

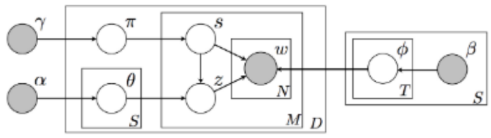

Fig. 3. Graphical representation of ASUM.

Here the approximate probability of sentiment j in comment d is π_{dj}, the approximate probability of aspect k for sentiment j in comment d is θ_{djk} and the approximate probability of word w in senti-aspect $\{k, j\}$ is ϕ_{jkw} is:

$$\pi_{dj} = \frac{C_{dj}^{DS} + \gamma_j}{\sum_{j'=1}^{S} C_{dj'}^{DS} + \gamma_{j'}} \quad (1)$$

$$\theta_{djk} = \frac{C_{djk}^{DST} + \alpha_{jk}}{\sum_{k'=1}^{T} C_{djk'}^{DST} + \alpha_{jk'}} \quad (2)$$

$$\phi_{jkw} = \frac{C_{jkw}^{STW} + \beta_{jw}}{\sum_{w'=1}^{V} C_{jkw'}^{STW} + \beta_{jw'}} \quad (3)$$

ASUM incorporates the use of *seed words* into the generative process. We use two sets of seed words, one per sentiment, containing positive and negative words that are not topic-specific, e.g., the sets of words can not be used to label a topic. The seed words were manually selected by the authors using the public list [22]. These words were chosen for their lack of sensitivity to context. Additionally we include the words, with the "not" prefix considered in the pre-processing step to present a semantic orientation. As further information we have the number of "stars" per comment, as ratings, to include. The use of the rating information and the seed words are used to initialize the generative process.

C. Step 3: User Comment Report

The goal of the last step is to provide meaningful feedback to developers. We therefore generate a report using JasperReports and iReport [1]. The report presents the classified information, organized by topics and sentiments, including 2 or 3 possible labels for each topic-sentiment. The labels are the words with the higher probability to belong to the topic-sentiment group. The seed words of each sentiment and a group of seed words with neutral meaning such as "recently" and "seen" are not used as possible labels. These words are rejected as they are not related to the application domain and thereby less useful for developers when revising the SRS. We evaluate the usefulness of the report in a human subject study as described below. The study helped answer important questions such as if the generated report saves developers time, if it is possible to extract the same information in less time and with less effort, and whether it is better to use the report compared to the original list of comments.

[1]Popular open source reporting engine and designer reporting software. More information on http://community.jaspersoft.com/

IV. Evaluation

The evaluation consists of two parts. First, we evaluate the information retrieval technique using standard measurements and second, we evaluate the quality of the generated report as a basis for new and changed requirements.

A. Evaluation of Approach

To evaluate the approach, we apply it on three different real-world data sets of user comments for publicly available software applications and we compare the results with two other ways of classifying topics (manual classification and similarity using the Jaccard coefficient). We choose to compare different classification schemes to experimentally analyze the strengths and weaknesses of our approach.

1) Experiment Setup: The experiment setup has two important steps: (1) the preparation of the data and (2) the classification of the data using different techniques for comparison.

2) Data Sets and Preprocessing: We extract user comments from the Android Market, for the following applications: Calorie Tracker, Mint.com Personal Finance, and Facebook for Android [2]. The selection of applications was based on the number of comments and the spread between positive and negative ratings. Also, we chose two free applications and one paid application. The thought behind this selection is that paid applications generate more negative comments than free applications. We manually classified topics for both Calorie Tracker and Mint.com, as the number of comments in both cases was manageable. We did not manually classify topics for the Facebook app as it had many more comments. We want to include Facebook in our analysis as an example of using the approach on a large data set. Table I shows the datasets used to evaluate our approach including the number of comments, sentences, and words.

The Calorie Tracker application is classified within the health and fitness category on the Android marketplace and it is a paid application. After preprocessing, the data set contains 327 comments, 874 sentences, and a wordlist of 865 elements as shown in Table I. The Mint.com Personal Finance from the Finance category and Facebook for Android from the social category are both free applications.

[2]More information about the applications on https://market.android.com/

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

TABLE I
DATA SETS OF USER COMMENTS.

Application	Comments	Sentences	Word List
Calorie tracker	327	874	865
Mint.com Personal Finance	383	579	1029
Facebook for Android	1941	3131	2592

3) Classification: We classify the data sets using three different techniques: manual classification (not Facebook), K-Median classification with the Jaccard coefficient of similarity, and ASUM.

The different classification techniques allow us to compare the results of different approaches to analyze the usefulness of each. Note that, regardless of classification technique, not all topics result in requirements changes. Some common topics could indicate bugs, and some topics with very few comments might not be important enough to cause changes. We exclude topics that only have one comment from our classifications.

a) Manual Classification: The manual classification of the comments for Calorie Tracker and Mint.com was conducted by the second author using the following process:

i) An user comment is analyzed and potential topics recorded on a separate sheet.
ii) As a comment can contain multiple sentences and topics, all topics present in a comment are recorded and each recorded topic is marked with the unique identifier of the comment from which it was derived.
iii) As user comments are in natural language and users do not use the same exact wording to express common ideas, the recorded topics are consolidated and separated based on semantics determined by the second author.

It took 13.6 hrs to classify the topics of the 327 comments of Calorie Tracker. Overall, the process was tedious and error-prone. In a second iteration of the classification procedure the second author reported 86 changes to the initial classification, indicating the difficulty of getting it right the first time around.

As we intend to use the manually classified data as truth sets, we would like to point out that they do not describe absolute truth as the second author is not a domain expert of either application, not involved in the development of either application, and reported that the process is error-prone. In fact, even within an organization, no two people would come up with the exact same classification of topics. This is an internal threat of validity.

To explore this potential weakness of the true set and minimize this threat, we conducted a small experiment to measure the difference in how people classify topics manually. We use this difference as a comparative measure to our automated technique. That is, if our automated technique is within the same range of recall, precision, and F-measure relative to the selected true set, the technique performs at least as well as manual classification (assuming that the manual classification is one acceptable solution), but with less effort. Recall is computed by the number of true positives divided by the sum of true positives and false negatives and precision is the number of true positives divided by the sum of true positives and false positives. The F-measure is computed by the precision and recall evenly weighted using the general form

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

In this experiment a true positive is a topic that matches with the true set, a false negative is a topic that is in the true set but not in the comparison set, and a false positive is a topic that is present in the comparison set but not in the true set.

Three graduate students were asked to classify a subset of the Calorie Tracker comments. The precision and recall compared to the true set, or Classification 0, are reported in Table II. It is worth noting that precision is held fairly steady whereas the recall measure shows great variability.

TABLE II
PRECISION, RECALL AND F-MEASURE OF MANUAL CLASSIFICATIONS.

Classification	Precision	Recall	F-Measure
Classification 1	70%	68%	68.98%
Classification 2	71%	48%	57.27%
Classification 3	75%	19%	30.32%

b) K-Median Clustering Classification: We also want to compare ASUM to a simpler automated technique. For this we chose K-Median clustering with the Jaccard similarity coefficient between pairwise sentences. The K-Median clustering with the Jaccard similarity was chosen because it is a common clustering technique to classify similar objects.

The K-Median classification is a variation of K-Means clustering, with the aim of trying to find the natural K cluster centers in a group of n topics by minimizing the distance between each topic to the center of the cluster to which it belongs. The difference is computing the median when the mean is not possible. With respect to the distances, we use the Jaccard similarity coefficient defined as the ratio between the size of the intersection and the size of the union of the sets (see Equation 5). Here, we compare the similarity of the 1-gram and 2-grams of each sentence. That is, we create a set for each sentence that contains the single words of the sentence (1-gram), and every consecutive two words (2-grams).

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{5}$$

K-Median clustering requires us to specify the number of clusters a priori where K corresponds to the number of expected topics in a data set. Our choices are somewhat subjective, as there is no standard way to determine optimal values. For our experiments, we chose different values of K for each application. For Calorie Tracker and Mint.com, the first value of K was chosen to match the number of manually extracted topics, the second value was chosen to double the number of manually extracted topics to account for the human factor of merging more topics based on semantics, and the third value of K was chosen to be much larger to see the effect of the algorithm on a large number of clusters. For Facebook, where we did not have a true set of topics, we chose two arbitrary K values to compare.

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

To minimize the human effort in inspecting topics, we chose the sentence representing the center of each cluster to classify the topic. The label of the cluster is thus the center sentence.

c) ASUM Classification: We run the adapted ASUM algorithm as described in Section III. ASUM provides us the approximate probabilities for each word within the extracted topics, we use the probabilities to assign a label per topic to reduce the amount of manual work required. To assign a label to each topic, we use the word with the highest probability. In doing this, we found that there are some topics that include a variety of words with the same probability. These groupings indicate a mix of actual topics and are excluded as misclassifications. Similarly to the other approaches, we also exclude topics that contain only one comment because these could be seen as comments in the original raw presentation.

As the wording of the generated labels in the three approaches will differ, we compare them on a semantic basis. For example, the manual classification could contain a topic named "sync problem", the K-Median could produce a topic labeled "the website does not sync", and ASUM could generate a topic labeled "sync". In our comparison, we consider these three topics the same as they semantically mean the same thing.

In ASUM, we also need to specify the number of topics prior to running the algorithm. For fair comparison, we use the same number of topics as in the K-Median experiments.

4) Experiments: Our evaluation compares the results of the three classification approaches on three different data sets of user comments from different mobile applications. For each experiment, we evaluate the topics discovered. There are three criteria for measuring the quality of the topics [15]: (1) the topics should be coherent (logical and consistent), (2) the topics should be specific enough to capture the details in the comments, and (3) the topics should be those that are discussed the most in the reviews. We make the assumption that the manually extracted classification (true set) adheres to these criteria as was checked with two careful iterations, and therefore compare the automatically derived topics to the true set. Another evaluation criterion, which is domain specific to RE, is that the derived topics should result in requirements changes. This aspect will be analyzed by inspection of the generated results.

B. Evaluation of Generated Report

To evaluate the quality of the generated report, we organized a human subject study with the goal of determining the usefulness of the information generated through our approach. In the study, participants were divided into two groups, a control group and a test group. The control group was given user comments without any pre-processing or classification and the test group was given the auto-generated report. In addition, all participants were given a Software Requirements Specification (SRS) for the Calorie Tracker application. As there are no SRSs distributed with the software on the Android Marketplace, the authors created the SRS based on their understanding of the application. The SRS contains a general description of the software application, 36 functional shall statements, 7 non-functional shall statements, a data model, and 6 use cases. The SRS was created without bias, that is without knowing the results of our approach. Furthermore, the SRS is not a utility in the study and remains the same for each group. All the participants had 1 hr to revise the SRS. The participants were asked to speak out loud and were interviewed after the experiments so that their thought process could be accurately captured and analyzed.

V. Experiment Results

A. Comparative Analysis

Table III shows the precision, recall, and F-measure for the K-Median and ASUM classifications in comparison to the true set for the Calorie Tracker application. For this experiment we used $K = 33, 66, 150$. We can see that the level of precision when using the K-Median approach stays stable over the different values of K. That means that the approach is not finding many topics that were not also found by the manual classification. The level of recall on the other hand increases significantly between $K = 33$ and $K = 66$, whereas it does not change much between $K = 66$ and $K = 150$. That means that the number of true positives increases a lot between $K = 33$ and $K = 66$, but only by a little as K gets bigger. The ASUM approach shows increasing precision with an increasing number of topics. As with K-Median, ASUM shows a similar trend with regard to the recall and F-measure. We can compare these numbers to the precision, recall, and F-measure numbers of different manual classifications presented in Table II. The F-measure shows how the performance in general increases and gives a general performance that reflects the levels of precision and recall for both classifications. Stable levels are achieved for bigger values of K. The manual results also show fairly stable precision levels, ranging from 70%-75%. Whereas the recall levels between different manual classifications range between 19%-68%.

TABLE III
CALORIE TRACKER. PERFORMANCE OF AUTOMATED TECHNIQUES COMPARED TO TRUE SET.

Classification	Precision	Recall	F-measure
K-Median Classification			
$K = 33$	77.77%	42.42%	54.6%
$K = 66$	82.14%	69.69%	75.5%
$K = 150$	87%	79%	82.8%
ASUM Classification			
$K = 33$	78%	21%	33.09%
$K = 66$	90%	81.81%	85.81%
$K = 150$	100%	78.78%	88.26%

Table IV shows the precision, recall, and F-measure of the K-Median and ASUM classifications compared to the true set for the Mint.com comments. For this experiment we used $K = 24, 48, 150$. As in the previous experiment, the results show similar trends with regard to precision and recall when using the K-Median approach. Again, the ASUM approach shows increasing numbers with regard to precision, recall, and F-measure with an increasing number of topics.

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
MINT.COM. PERFORMANCE OF AUTOMATED TECHNIQUES COMPARED TO TRUE SET.

Classification	Precision	Recall	F-measure
K-Median Classification			
$K = 24$	90%	37.5%	53.43%
$K = 48$	93.75%	62.5%	75.43%
$K = 150$	94.11%	66.67%	78.24%
ASUM Classification			
$K = 24$	62.5%	20.83%	31.44%
$K = 48$	86.67%	54.16%	66.64%
$K = 150$	90%	75%	80%

As there is no true set for the Facebook data, we discuss the results qualitatively. We ran K-Median and ASUM with $K = 70$ and $K = 100$. Figures 4 and 5 show word clouds of the results for $K = 70$. Each sentence in the word clouds represents a "topic". In the word cloud for the K-Median approach the sentences are the centers of each cluster. The visual size of the sentence is determined by the number of comments that belong to the cluster. In a way, that could represent the significance of the topic, if we assume that the more a topic is mentioned by users, the more important it is. In the word cloud for the ASUM approach, on the other hand, the sentences are selected by looking at the probabilities of different sentences in combination with hand-picking a sentence among the ones with probabilities between 0.99-1.0.

In both cases, during this process we discovered groups that contain mixed topics, as well as groups of duplicated topics. The duplicates are visible in the word clouds. For example the word cloud for K-Median shows a topic called "keeps force close" and another topic called "force closes a lot". The mixed topics cannot be discovered in this representation. Even with mixed topics and duplicates, the word clouds do provide useful information. It is not difficult to pick out five to ten topics in each that would inspire requirements changes.

Comparing the two word clouds we can see that the distribution of comments per topic is more even for ASUM than for K-Median. With a low K value relative to the number of comments, ASUM does not produce very good results. That is, many of the groups contain mixed topics.

B. Reasoning about Requirements

To provide an example of the kinds of topics present in the user comments, Table V shows ten topics per application. For Calorie Tracker and Mint.com the topics were extracted manually, whereas for Facebook the topics were generated using ASUM. We can see that the topics are relevant in the sense that they cover important features of these applications. However, we still need the sentiment, or a representative comment to understand what users think of each topic.

We can provide a representative comment for each topic to provide a requirements engineer with some context on the topic. Tables VI, VII, and VIII show three topics for each of the three applications, and provide sample of sentences for each of the topics. A requirements engineer can look at a short list of topics with a short list of representative user comments and decide whether or not the topic is a candidate for a requirements change. For example, the topic "Features" extracted from the Facebook comments indicates that the requirements should include more of the features available on other platforms. Whereas, the topic "Updates" for the same application indicates a potential bug fix. A requirements engineer could easily sort through this short list more efficiently than when having to consider the whole set of comments. In addition to useful sentences under each topic, we also found many nonsense sentences. This could happen when a sentence is supposed to be considered together with previous ones, or when a user writes poor sentences.

TABLE V
SAMPLE TOPICS PER APPLICATION.

Calorie Tracker	Mint.com	Facebook
Barcode scanner	Alerts	Updates
Calorie counter	International use	Features
Database	Online version	Developers
Not enough features	Platform	Messages
Creating an account	Update	Force closes
Sign in/log in	Usability	Photos
Bad interface	Transaction categories	Friends
Wrong visualization of food	Duplicates	Chat
Difficult to use	Budget	Notifications
Break down of nutrition	Cash transactions	Sony Ericson

TABLE VI
CALORIE TRACKER APPLICATION SENTENCES

Topic: *Barcode scanner*
Works fine, but if it had a barcode scanner it would be much better. Barcode scanner still not working.
Topic: *Database*
It doesn't have a large food database and there isn't a way to manually input. Very easy to use, huge food database.
Topic: *Sign in/log in*
Took seven attempts to create account. Wont let me create an account.

TABLE VII
MINT APPLICATION SENTENCES

Topic: *Usability*
Been having problems with it since installed which kind of makes it useless. This is useless if your bank is not listed.
Topic: *Transaction categories*
However I would like to have a way to look at the most recent transaction. Only allows one category per transaction.
Topic: *Duplicates*
Duplicate accounts are still an issue. Still no fix for duplicate entries.

C. Generated Report Analysis

The human study was conducted with participants in two groups, a control and a test group. Threats to external validity include to what degree the participants of the study are representative of developer teams. The participants are undergraduate and graduate students of the Computer Science and Electrical & Computer Engineering Departments at the University of Delaware and are familiar with application development and software requirements. Each group had seven participants revise the SRS of the Calorie Tracker application

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

Fig. 4. Graphical representation of K-Median on Facebook comments.

Fig. 5. Graphical representation of ASUM on Facebook comments.

TABLE VIII
FACEBOOK APPLICATION SENTENCES

Topic: ***Updates***
Glitches, constantly refreshes and goes back to top of page. The last update messed up the newsfeed.
Topic: ***Features***
The features on this app should be like the desktop, its too many things you cant do. Still missing some features that the iphone has but pretty good overall.
Topic: ***Developers***
I hope the devs can look into this and see about adding a rotate features. You need to allow us to log out.

TABLE IX
STATISTICS FROM THE STUDY.

	Control group	**Test group**
Comments analyzed	250	376
Total time (min)	46	44
Information recorded		
# New Requirements	8	12
# Req. Changes (alterations, reprioritization and deletions)	2	4
# Total Changes	10	16

using either the raw user feedback (control group) or the generated report (test group). The selection of participants for each group was random and the graduate and undergraduate students were evenly split to avoid bias. The background of all the participants is almost the same and this information was obtained with previous background questions. To ensure that the results are applicable to development teams, additional to the experience with software requirements, some participants are students whose major is software engineering. The software engineering majors were also split between the test and control groups. A separate comparison showed that the behavior and results were similar compared with those obtained by the other participants. Threats to internal validity include the selection of conditions to generate the best results. To minimize this threat, the conditions for the experiment were the same for all the participants, a quiet place (environment), the same material, time, instructions, and procedure. We recorded the number of comments analyzed and also the time spent on the task.

Table IX presents the average of the results per group as well as some statistics, including the average number of requirements that were added, altered, deleted, and reprioritized. In the ideal case, every SRS revision should be evaluated for correctness and consistency. In this study, we assume every revision to be correct and counted toward the average. The numbers were always higher for the test group than for the control group. An obvious reason is that the test group had less comments to read through with them organized through the automatic classification. In the interview the participants of the test group said that the classified presentation of the comments makes the process fast.

Figure 6 shows some of the topics in the SRS and the average revisions made per topic for each of the groups. The number of changes made by the test group using the generated report is superior for topics like views, platform, usability, and accounts and is comparable to the control group in other topics like food log, database, reliability, and security. The comparison shows an advantage of using our approach to generate a report of topics to revise the SRS. It is also important to mention here that the results of the test group were obtained analyzing all the comments in the generated report instead of the, on average, 250 comments analyzed by the control group.

To study if these results are in fact significant, we perform the t-test to evaluate whether or not the mean of revised material

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

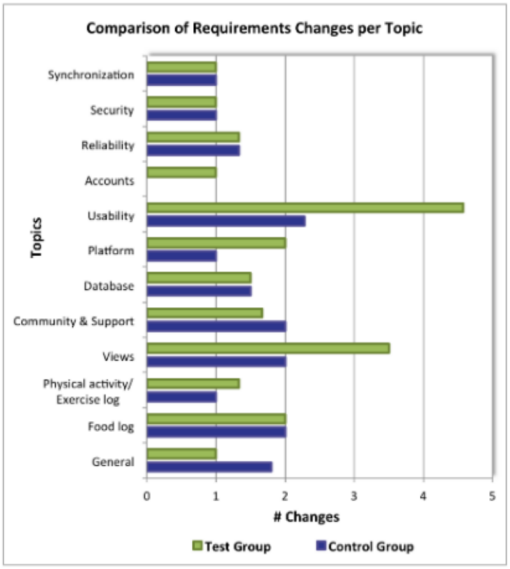

Fig. 6. Comparison of changes in topics for each group of the study.

in the SRS and the number of comments analyzed by the test group are significantly higher than for the control group. The null hypothesis of this study is stated as $H_0 : \mu_{test} <= \mu_{control}$, i.e, the mean of the test group is lower than or equal to the mean of the control group. The value of p is used to determine its likelihood. If deemed unlikely, then the alternative hypothesis ($H_a : \mu_{test} > \mu_{control}$) is accepted, i.e., the mean of the test group is higher than the mean of the control group, and that it is unlikely due to chance. The null hypothesis is rejected if the value of p is lower than that determined by a level of significance. The level of significance used in this analysis is 0.05.

For the case of the number of comments analyzed, the p-value is 0.023, and for the number of revisions the p-value is 0.0057. As both values are lower than the level of significance, the null hypothesis is rejected and consequently the alternative hypothesis accepted. This permits us to conclude the utility of the report generated by our approach.

We were also interested in studying the process participants use when analyzing the user comments to make decisions about requirements revisions. As we did not provide guidelines in how to use the information provided, each participant chose their own strategy for the task. We asked the participants at the end of the revision about the strategies used and how useful they were to complete the process. The most common strategy in the control group was to go through each comment and to select those with useful information. The strategies used by the test group varied more. Some used the same strategy as the control group and others chose to read by topic-sentiment cluster and then make the revisions. Some of the participants were comfortable using the labels of the groups of comments to assign or try to match the comments, with requirements categories, others did not use this additional information.

Finally, we asked the participants the following questions: *How easy was it to use the feedback to change the requirements?* and *How useful do you think this feedback is to software development?*. Figure 7 summarizes the answers to these questions. The effort was perceived higher for the control group highlighting the advantages of the automated classification of user comments. On the other hand, the utility of the information is higher for both groups than we expected, which shows the importance of user feedback for developers, and confirms the need for better techniques to accomplish this.

Fig. 7. Quality Indicators.

D. Discussion of Results

The results by both K-Median/Jaccard and ASUM look promising. Compared to manual classification, the results show that a good representation of topics can be extracted in a small amount of time compared to the manual process.

There are however some limitations of the K-Median approach that have not yet been discussed. Because we only consider the center sentence of each cluster to represent the group, without exploring whether or not the rest of the sentences in the cluster also describe the same topic, we do not catch mis-classifications. In ASUM, we use the probabilities of the words to classify the group, mis-classifications are therefore more apparent as several words can have the same probability, i.e., there is no clear label for the topic. The ASUM approach thereby imposes a stricter classification on whether or not to count a topic as a true positive. The results of K-Median are thus over-optimistic in comparison to ASUM.

Another problem with the K-Median approach is that not only could there be mixed topics, but there are many duplicated topics, which is reflected in the recall levels. Even with larger numbers of K, rather than finding the missing topics, more duplicate clusters are formed.

A limitation of ASUM is that it requires a fairly large number of topics to avoid mis-classification. This is clearly shown in the word cloud of the Facebook comments, as the even distribution among the comments in the topics is visible in the font size. A larger number of topics is obviously less ideal for the analyst that has to sift through the generated topics. When using a larger K value, ASUM performs better and the number of comments per topic is less evenly distributed.

Determining the ideal value of topics, or K, using either of the automated approaches is challenging. However, our experiments showed an interesting trend, which led to the hypothesis that the number of topics present in a collection of user comments is not proportional to the number of comments. It seems that the number of actual topics levels off.

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.

Furthermore, a company might be interested in only the top 10 or 20 topics for the next release, and the K value could thus be chosen accordingly (keeping in mind that the K needs to be bigger - at least twice the number of topics - than the intended number of topics due to duplication). Choosing a large K for the ASUM approach could yield good results as noticed in the Facebook experiment. However, going through a large number of topics, such as 100-200, is still a tedious task for a requirements engineer, although the number is reduced from the original 1941 comments.

We believe ASUM shows great promise because we can easily eliminate clusters that contain mixed topics leaving only classified information. Furthermore, as a company could be interested in a smaller number of topics, we can use the ASUM approach with a large K, then rank the topics according to a quality criterion that considers word-spread probabilities among other things. Finally, ASUM's sentiment classification, provides requirements engineers with the information of which topics are positive and which topics are negative. Although this part is not detailed in this paper, it was used to present the topic-sentiment clusters in the user comment report, proving to be useful for the test group in order to suggest changes in requirements.

VI. Conclusion and Future Work

This paper presents an initial assessment of the validity of using topic modeling to analyze a set of user comments for topics relevant to requirements evolution. We adapted and applied a topic modeling technique on three different sets of user comments for mobile applications, and could see that the approach clearly generates a good representation of topics that could be relevant with regard to requirements changes.

Although the results are promising, this work also serves as a motivation for future improvements in the technique. We are exploring the sentiment analysis to automatically extract user opinions on the topics extracted, as a way to provide requirements engineers additional information about each extracted topic. We are currently working on enhancing ASUM's sentiment analysis scheme to also include user ratings associated with the comments in the calculation of the sentiment value and not just as initial information of the generative process.

One way we could address the mis-classification, or mix of topics, that results from running ASUM with a low number of topics, is to apply techniques to avoid groupings of irrelevant words. We have already filtered out some of the irrelevant words using a stop words list, but there are more domain specific words such as "good" or "needs", that occur in many comments, but should not be used as classifiers. One approach could be to use Part-Of-Speech Tagging (POST) to identify nouns and verbs, and give them a higher weight in the classification than for example adjectives.

Another interesting improvement involves the design of quality criteria to evaluate the quality of each topic generated. A high quality cluster would be one that has low levels of mis-classification and is not a duplicate of a previously generated topic. We are looking for ways to computationally express these qualities and use them to rank the topics generated. By doing so, we can for example easily extract the top 10 topics, or set a threshold based on the quality, to avoid presenting irrelevant results to the requirements engineer.

References

[1] Hailong Wang, Juan Li, Qing Wang, and Ye Yang. Quantitative Analysis of Requirements Evolution across Multiple Versions of an Industrial Software Product. *Asia Pacific Softw. Eng. Conference*, Nov 2010.

[2] K. Schneider. Focusing spontaneous feedback to support system evolution. In *Proceedings of the Requirements Engineering Conference (RE'11)*, pages 165–174, 2011.

[3] Daniel Ramage, Susan T. Dumais, and Daniel J. Liebling. Characterizing microblogs with topic models. In *ICWSM*, 2010.

[4] Stefan Siersdorfer, Sergiu Chelaru, and Via Augusta. How Useful are Your Comments Analyzing and Predicting YouTube Comments and Comment Ratings. *International World Wide Web Conference Committee (IW3C2)*, pages 891–900, 2010.

[5] Hazeline U. Asuncion, Arthur U. Asuncion, and Richard N. Taylor. Software traceability with topic modeling. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE*, pages 95–104, 2010.

[6] J.I. Maletic and A. Marcus. Supporting program comprehension using semantic and structural information. *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, pages 103–112.

[7] Abram Hindle, Michael W. Godfrey, and Richard C. Holt. What's hot and what's not: Windowed developer topic analysis. *International Conference on Software Maintenance*, pages 339–348, September 2009.

[8] Denys Poshyvanyk and Andrian Marcus. Using information retrieval to support design of incremental change of software. *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*, page 563, 2007.

[9] Giuseppe Scanniello and Andrian Marcus. Clustering Support for Static Concept Location in Source Code. *2011 IEEE 19th International Conference on Program Comprehension*, pages 1–10, June 2011.

[10] Bogdan Dit, Denys Poshyvanyk, and Andrian Marcus. Measuring the Semantic Similarity of Comments in Bug Reports. *Science*.

[11] Huiying Li, Li Zhang, Lin Zhang, and Jufang Shen. A User Satisfaction Analysis Approach for Software Evolution. *Conference on Progress in Informatics and Computing (PIC), 2010 IEEE International*, (2007):1093–1097, 2010.

[12] Changsheng Liu, Yanzhen Zou, Sibo Cai, Bing Xie, and Hong Mei. Finding the Merits and Drawbacks of Software Resources from Comments. *Analysis*, pages 432–435, 2011.

[13] Samuel Brody. An unsupervised aspect-sentiment model for online reviews. *Annual Conference of Technologies*, (Jun.):804–812, 2010.

[14] Jayanta Kumar Pal and Abhisek Saha. Identifying themes in social media and detecting sentiments. In *International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 452–457. IEEE Computer Society, 2010.

[15] Yohan Jo and A.H. Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.

[16] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, page 375, 2009.

[17] Michael W. Berry and Jacob Kogan. *Text Mining Applications and theory*. John Wiley & Sons, 2010.

[18] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101 Suppl 1:5228–35, April 2004.

[19] S. R. Das and M. Y. Chen. Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web. *Management Science*, 53(9):1375–1388, Sep. 2007.

[20] David M Blei. Introduction to Probabilistic Topic Models. *Communications of the ACM*, pages 1–16, 2012.

[21] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[22] Minqing Hu, Bing Liu, and South Morgan Street. Mining and Summarizing Customer Reviews. *Text*, 2004.

Authorized licensed use limited to: DUBLIN CITY UNIVERSITY. Downloaded on June 01,2020 at 06:29:16 UTC from IEEE Xplore. Restrictions apply.